CSC 311
Digital Computer Logic Design
(Lecture 2)

Credits:

Slides adapted from:

J.F. Wakerly, *Digital Design*, 4/e, Prentice Hall, 2006

C.H. Roth, *Fundamentals of Logic Design*, 5/e, Thomson, 2004

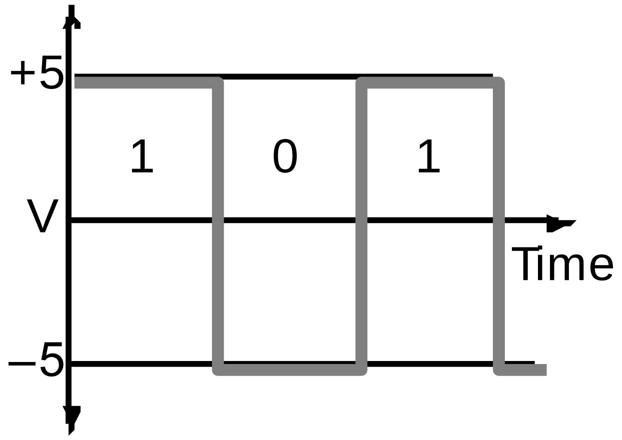
A.B. Marcovitz, *Intro. to Logic and Computer Design*, McGraw Hill, 2008

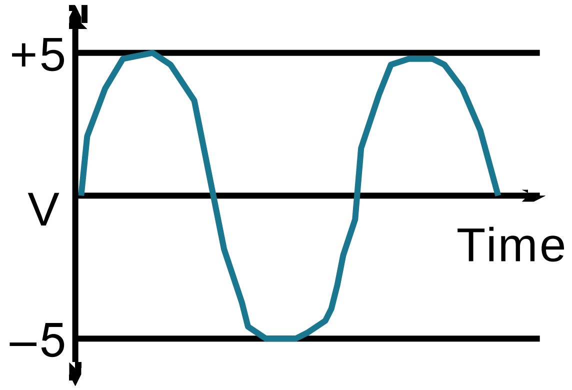
R.H. Katz, G. Borriello, *Contemporary Logic Design*, 2/e, Prentice-Hall, 2005

- Introduction to Basics: Logic Gates
- Combinational Circuits (Part)

Introduction: Digital Systems

Digital vs. Analog Waveforms





Digital:
only assumes discrete values

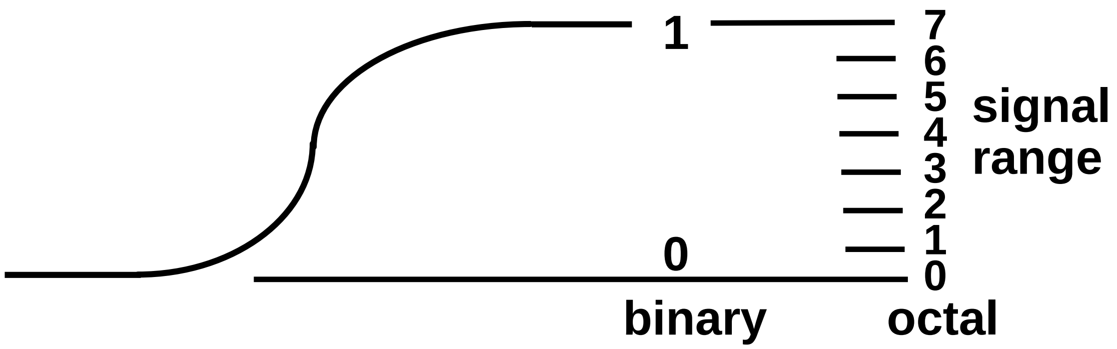
Analog:
values vary over a broad range continuously

LOGIC GATES

Digital Computers

- Imply that the computer deals with digital information, i.e., it deals with the information that is represented by binary digits
- Why *BINARY* ? instead of Decimal or other number system ?

* Consider electronic signal



* Consider the calculation cost - Add

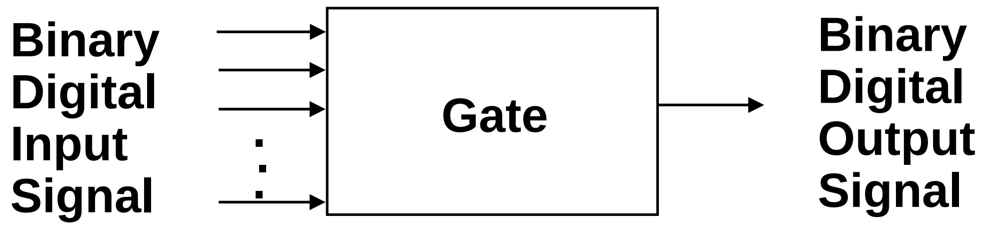
	0	1
0	0	1
1	1	10

	0	1	2	3	4	5	6	7	8	9
0	0	1	2	3	4	5	6	7	8	9
1	1	2	3	4	5	6	7	8	9	10
2	2	3	4	5	6	7	8	9	10	11
3	3	4	5	6	7	8	9	10	11	12
4	4	5	6	7	8	9	10	11	12	13
5	5	6	7	8	9	10	11	12	13	14
6	6	7	8	9	10	11	12	13	14	15
7	7	8	9	10	11	12	13	14	15	16
8	8	9	10	11	12	13	14	15	16	17
9	9	10	11	12	13	14	15	16	17	18

Logic Gates

- The building blocks used to create digital circuits are **logic gates**
- There are three elementary logic gates and a range of other simple gates
- Each gate has its own **logic symbol** which allows complex functions to be represented by a logic diagram
- The function of each gate can be represented by a **truth table** or using **Boolean notation**

LOGIC GATE



Types of Basic Logic Blocks

- **Combinational Logic Block**
  Logic Blocks whose output logic value depends only on the input logic values

- **Sequential Logic Block**
  Logic Blocks whose output logic value depends on the input values and the state (stored information) of the blocks

Functions of Gates can be described by

- **Logic Diagrams**
- **Truth Table**
- **Boolean Function**
- **Karnaugh Map**

How do we describe the behavior of gates

- Logic diagrams

A graphical representation of a circuit; each gate has its own symbol

- Truth tables

A table showing all possible input values and the associated output values

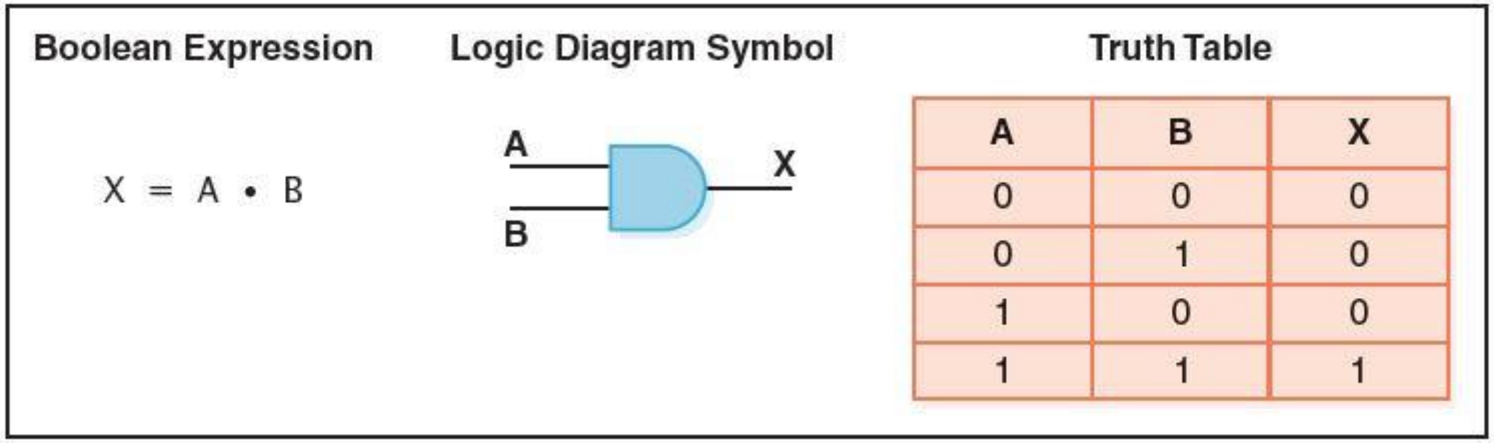
- Boolean expressions

Uses Boolean algebra, a mathematical notation for expressing two-valued logic

AND Gate

An AND gate accepts two input signals

If both are 1, the output is 1; otherwise, the output is 0


Boolean Expression

$X = A \cdot B$

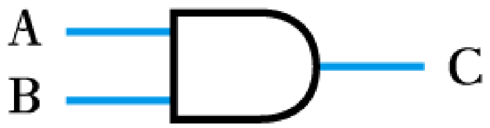
Logic Diagram Symbol

Truth Table

A	B	X
0	0	0
0	1	0
1	0	0
1	1	1

FIGURE 4.2 Representations of an AND gate

- **The AND gate**



(a) Circuit symbol

A	B	C
0	0	0
0	1	0
1	0	0
1	1	1

(b) Truth table

$C = A \cdot B$

(c) Boolean expression

OR Gate

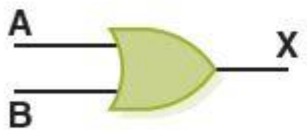
An OR gate accepts two input signals

If both are 0, the output is 0; otherwise, the output is 1

Boolean Expression

X = A + B

Logic Diagram Symbol

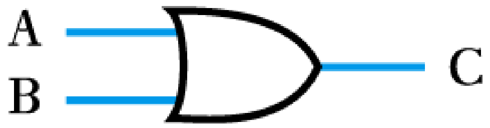


Truth Table

A	B	X
0	0	0
0	1	1
1	0	1
1	1	1

FIGURE 4.3 Representations of an OR gate

- **The OR gate**



(a) Circuit symbol

A	B	C
0	0	0
0	1	1
1	0	1
1	1	1

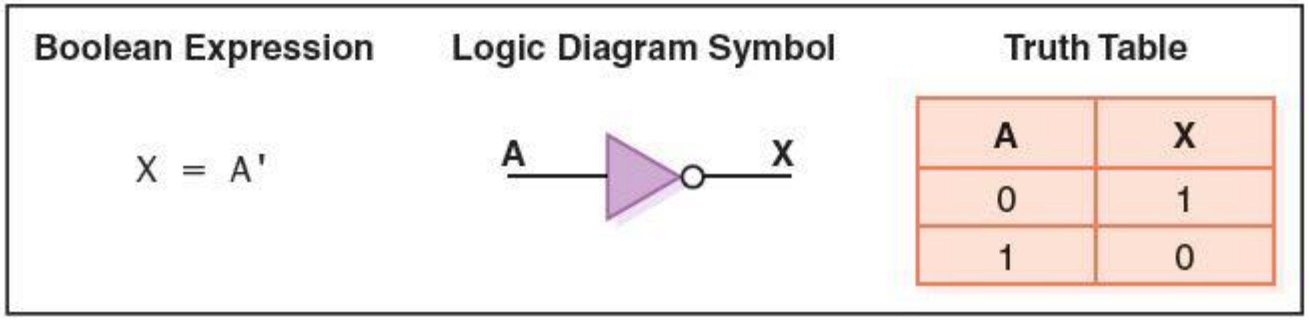
(b) Truth table

$$C = A + B$$

(c) Boolean expression

NOT Gate

A NOT gate accepts one input signal (0 or 1) and returns the complementary (opposite) signal as output


Boolean Expression

$X = A'$

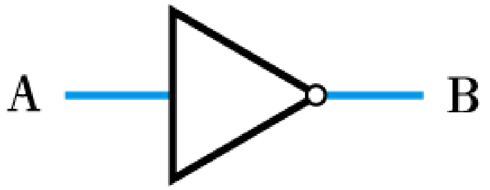
Logic Diagram Symbol

Truth Table

A	X
0	1
1	0

FIGURE 4.1 Representations of a NOT gate

- **The NOT gate (or inverter)**



(a) Circuit symbol

A	B
0	1
1	0

(b) Truth table

$$B = \overline{A}$$

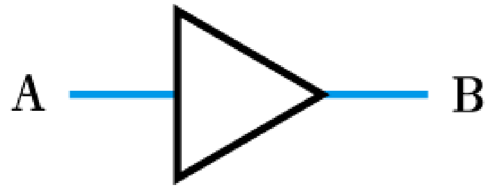
(c) Boolean expression

A logic buffer gate

A Buffer gate accepts one input signal (0 or 1) and returns the same signal as output

A Buffer has only one Input, its Output follows the same Logic State as the Input. Used as delay element in Digital Electronics. Inverter of inverter.

- **A logic buffer gate**



(a) Circuit symbol

A	B
0	0
1	1

(b) Truth table

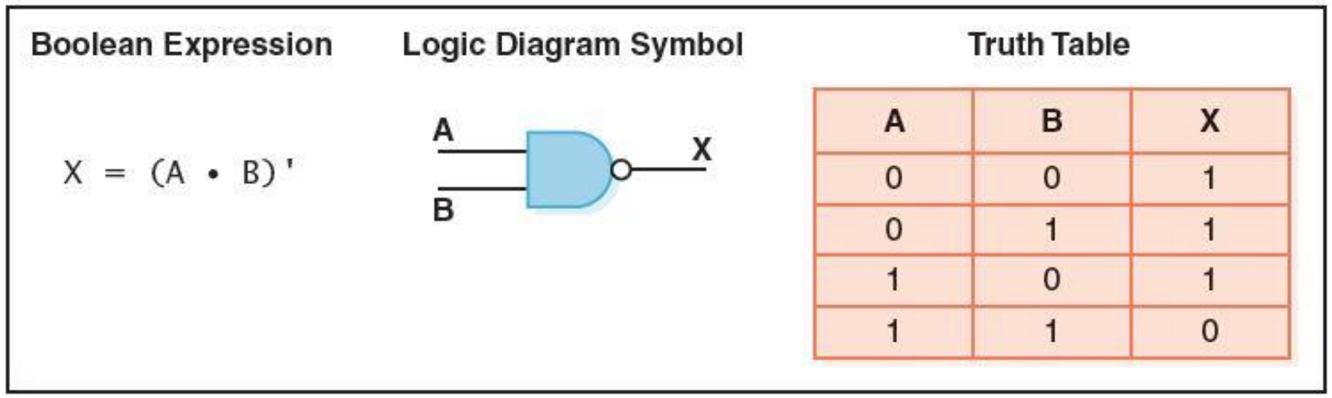
$B = A$

(c) Boolean expression

NAND Gate

The NAND (“NOT of AND”) gate accepts two input signals

If both are 1, the output is 0; otherwise, the output is 1

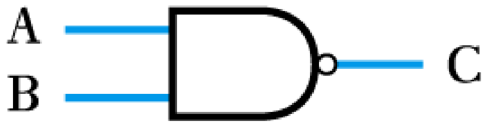


$X = (A \cdot B)'$

A	B	X
0	0	1
0	1	1
1	0	1
1	1	0

FIGURE 4.5 Representations of a NAND gate

- **The NAND gate**



(a) Circuit symbol

A	B	C
0	0	1
0	1	1
1	0	1
1	1	0

(b) Truth table

$$C = \overline{A \cdot B}$$

(c) Boolean expression

NOR Gate

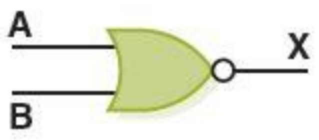
The NOR (“NOT of OR”) gate accepts two inputs

If both are 0, the output is 1; otherwise, the output is 0

Boolean Expression

$X = (A + B)'$

Logic Diagram Symbol

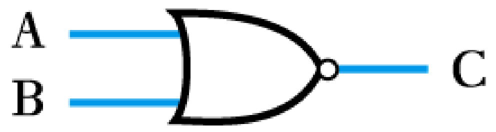


Truth Table

A	B	X
0	0	1
0	1	0
1	0	0
1	1	0

FIGURE 4.6 Representations of a NOR gate

- **The NOR gate**



(a) Circuit symbol

A	B	C
0	0	1
0	1	0
1	0	0
1	1	0

(b) Truth table

$$C = \overline{A + B}$$

(c) Boolean expression

XOR Gate (The Exclusive OR)

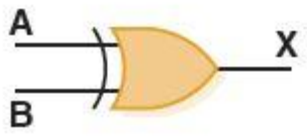
An XOR gate accepts two input signals

If both are the same, the output is 0; otherwise, the output is 1

Boolean Expression

$X = A \oplus B$

Logic Diagram Symbol

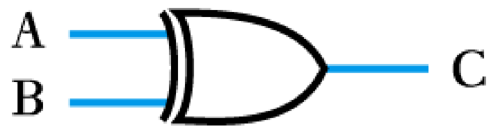


Truth Table

A	B	X
0	0	0
0	1	1
1	0	1
1	1	0

FIGURE 4.4 Representations of an XOR gate

- **The Exclusive OR gate**



(a) Circuit symbol

A	B	C
0	0	0
0	1	1
1	0	1
1	1	0

(b) Truth table

$$C = A \oplus B$$

(c) Boolean expression

XOR Gate

Note the difference between the XOR gate and the OR gate; they differ only in one input situation

When both input signals are 1, the OR gate produces a 1 and the XOR produces a 0

XOR is called the *exclusive OR* because its output is 1 if (and only if):

- *either* one input *or* the other is 1

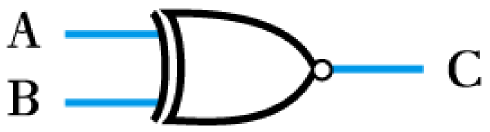
XNOR Gate (The Exclusive NOR)

An XNOR gate accepts two input signals

If both are the same (similar), the output is 1;

otherwise, the output is 0

- **The Exclusive NOR gate**



(a) Circuit symbol

A	B	C
0	0	1
0	1	0
1	0	0
1	1	1

(b) Truth table

$$C = \overline{A \oplus B}$$

(c) Boolean expression

Gates with More Inputs

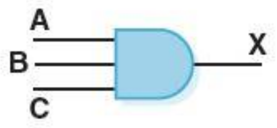
Some gates can be generalized to accept three or more input values

A three-input AND gate, for example, produces an output of 1 only if all input values are 1

Boolean Expression

$X = A \cdot B \cdot C$

Logic Diagram Symbol



Truth Table

A	B	C	X
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

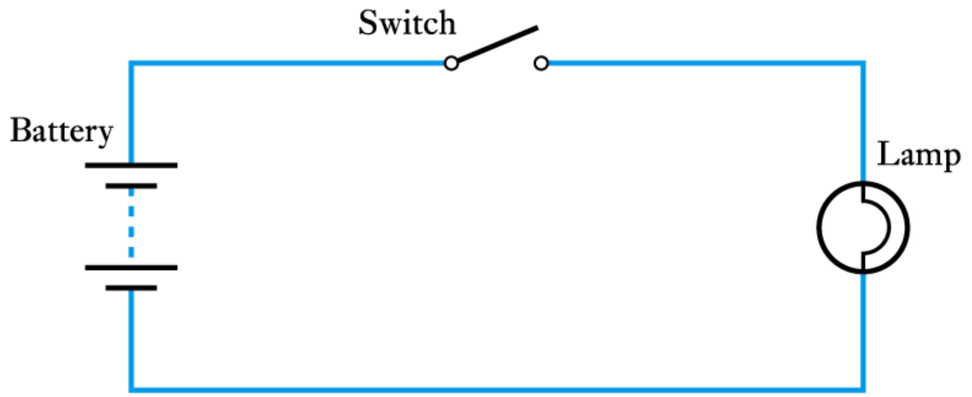
FIGURE 4.7 Representations of a three-input AND gate

Review of Gate Processing

Gate	Behavior
NOT	Inverts its single input
AND	Produces 1 if all input values are 1
OR	Produces 0 if all input values are 0
XOR	Produces 0 if both input values are the same
XNOR	Produces 1 if both input values are the same
NAND	Produces 0 if all input values are 1
NOR	Produces 1 if all input values are 0

Representations Using Switches

- A **binary quantity** is one that can take only 2 states



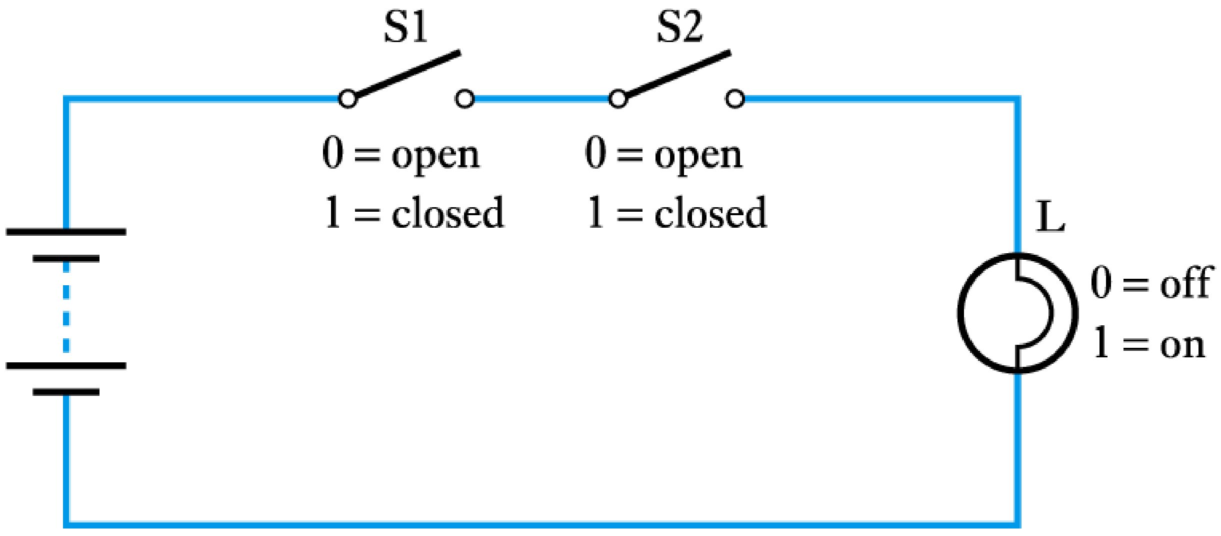
A simple binary arrangement

S	*L*
OPEN	OFF
CLOSED	ON

S	*L*
0	0
1	1

A truth table

- A binary arrangement with two switches in series



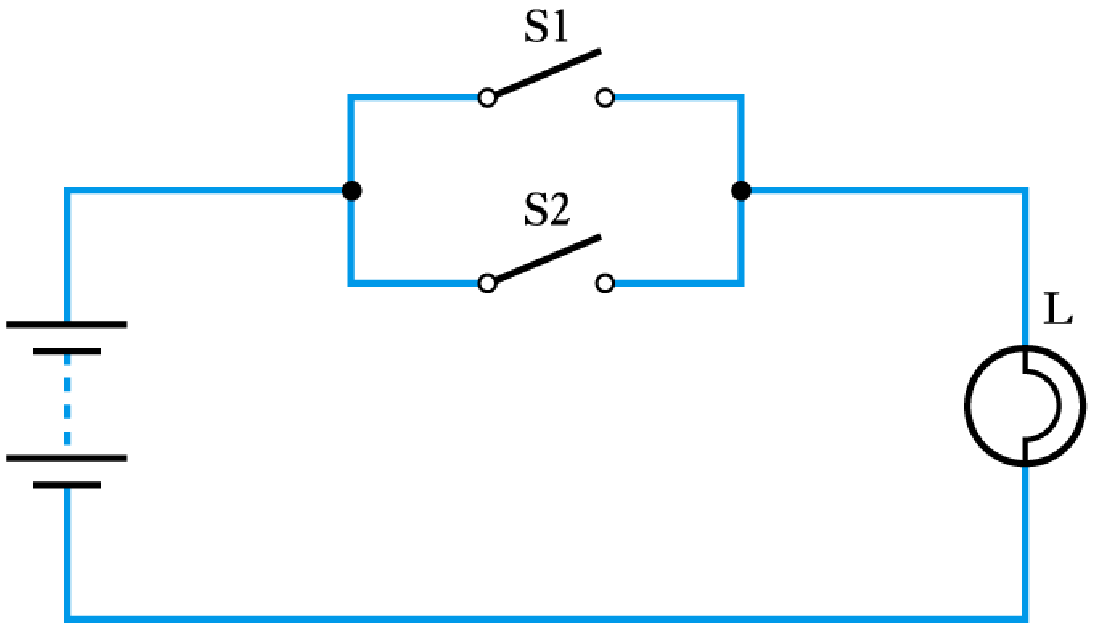
(a) Circuit

S1	S2	L
0	0	0
0	1	0
1	0	0
1	1	1

(b) Truth table

L = *S1* AND *S2*

- A binary arrangement with two switches in parallel



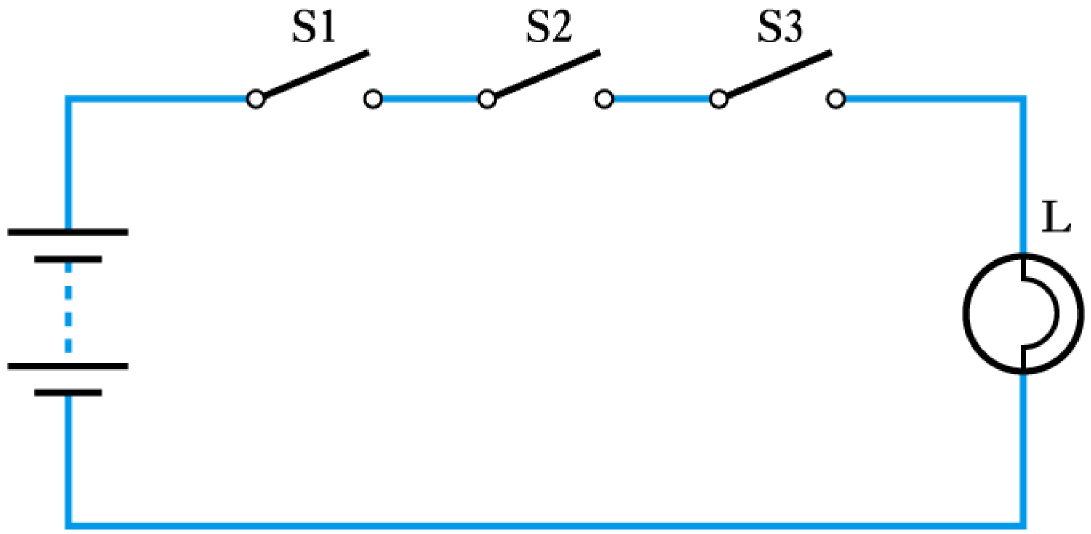
(a) Circuit

S1	S2	L
0	0	0
0	1	1
1	0	1
1	1	1

(b) Truth table

$L = S1$ OR $S2$

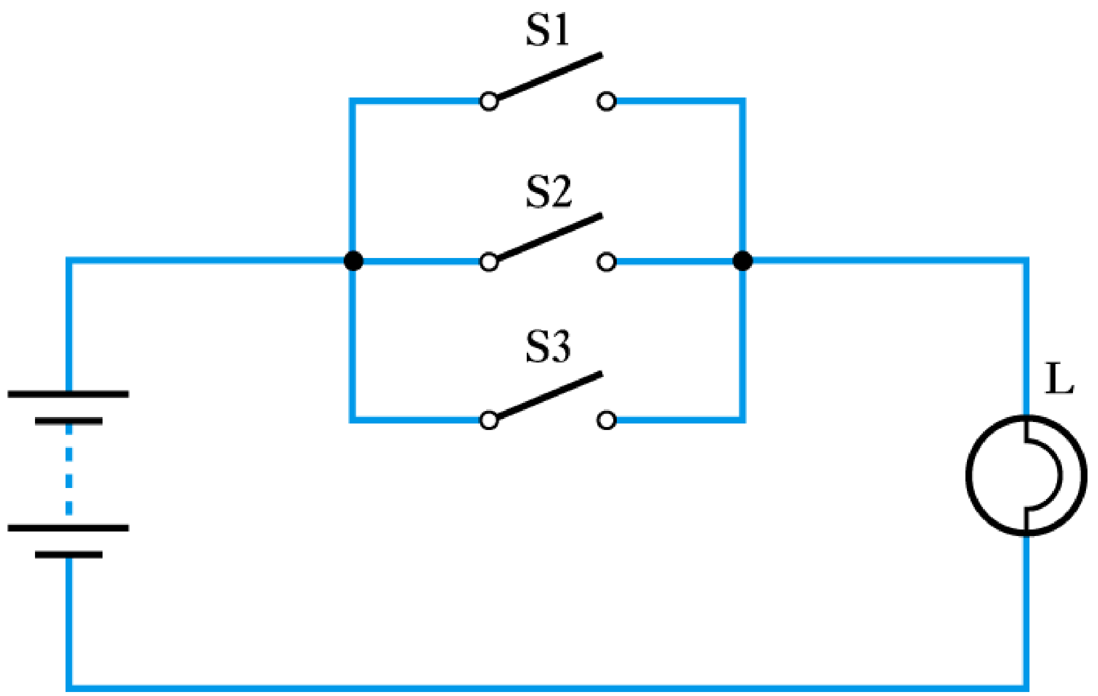
- Three switches in series



S1	S2	S3	L
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

L = *S1* AND *S2* AND *S3*

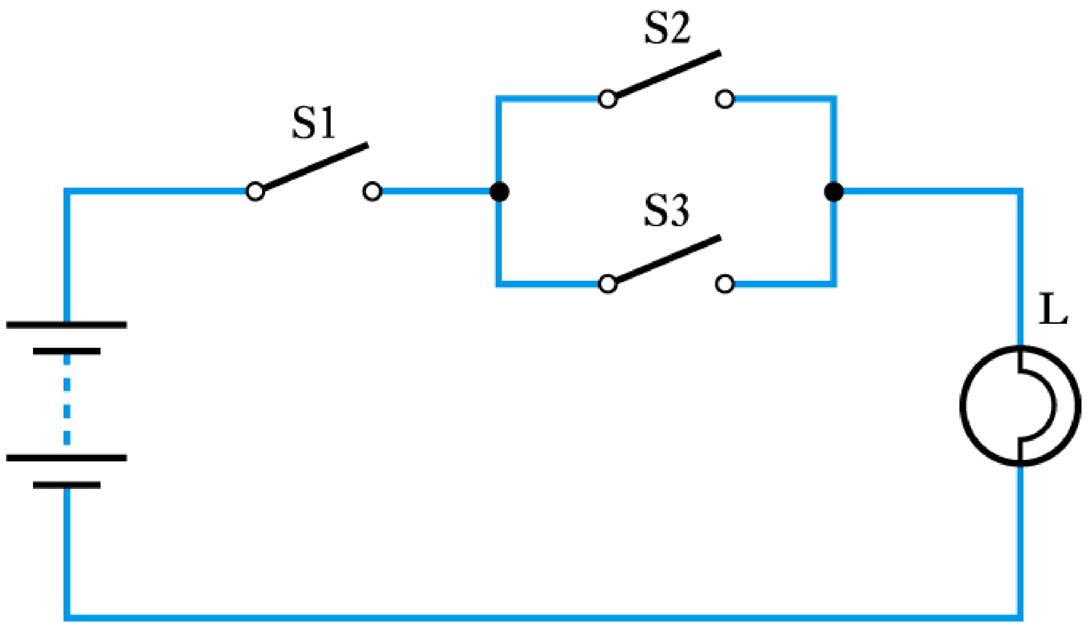
- Three switches in parallel



S1	S2	S3	L
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

L = *S1* OR *S2* OR *S3*

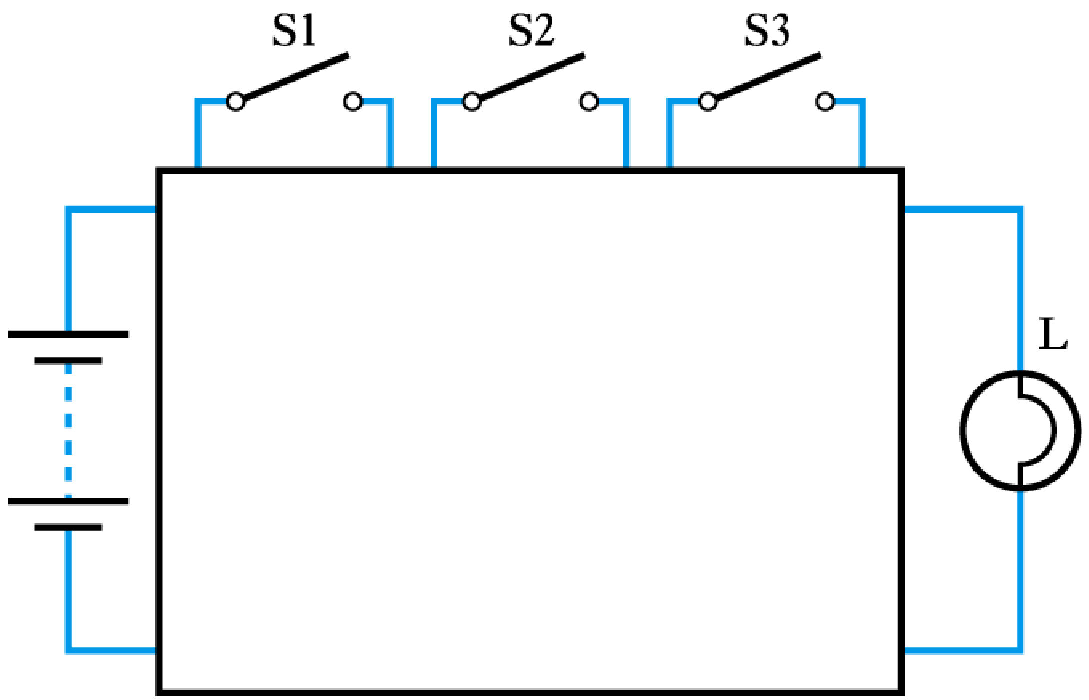
- A series/parallel arrangement

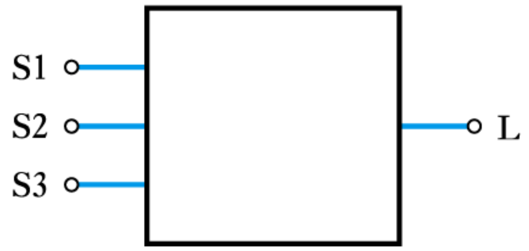


S1	S2	S3	L
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

L = *S1* AND (*S2* OR *S3*)

- Representing an unknown network



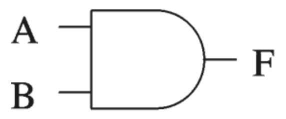


SUMMARY

- Hardware consists of a few simple building blocks
  - These are called *logic gates*
    - AND, OR, NOT, …
    - NAND, NOR, XOR, …
- Logic gates are built using transistors
    - NOT gate can be implemented by a single transistor
    - AND gate requires 3 transistors
- Transistors are the fundamental devices
    - Pentium consists of 3 million transistors
    - Compaq Alpha consists of 9 million transistors
    - Now we can build chips with more than 100 million transistors

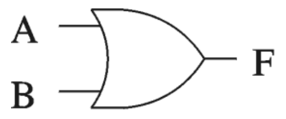
Basic Concepts

- Simple gates
  - AND
  - OR
  - NOT
- Functionality can be expressed by a truth table
  - A truth table lists output for each possible input combination



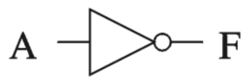
AND gate

A	B	F
0	0	0
0	1	0
1	0	0
1	1	1



OR gate

A	B	F
0	0	0
0	1	1
1	0	1
1	1	1



NOT gate

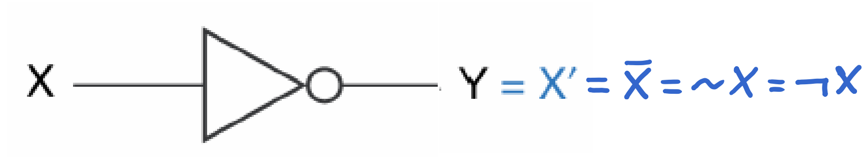
A	F
0	1
1	0

Logic symbol

Truth table

NOT, AND, OR

Truth Tables:

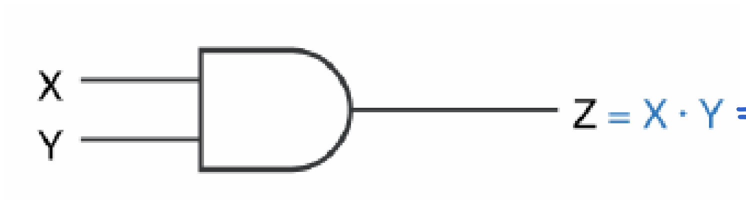


$Y = X' = \bar{X} = \sim X = \neg X$

NOT gate

X	Y
0	1
1	0

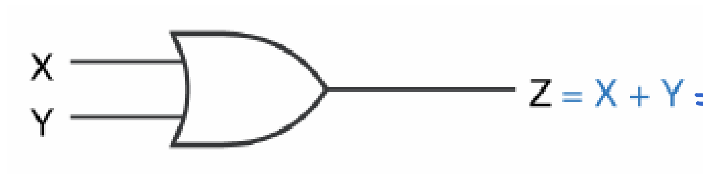
The NOT function is also known as INVERTER or COMPLEMENT



$Z = X \cdot Y = X \wedge Y$

AND gate

X Y	Z
0 0	0
0 1	0
1 0	0
1 1	1



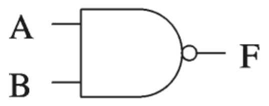
$Z = X + Y = X \vee Y$

OR gate

X Y	Z
0 0	0
0 1	1
1 0	1
1 1	1

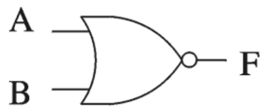
Basic Concepts (cont.)

- Additional useful gates
  - NAND
  - NOR
  - XOR
- NAND = AND + NOT
- NOR = OR + NOT
- XOR implements exclusive-OR function



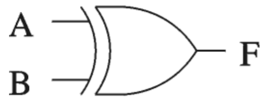
NAND gate

A	B	F
0	0	1
0	1	1
1	0	1
1	1	0



NOR gate

A	B	F
0	0	1
0	1	0
1	0	0
1	1	0



XOR gate

A	B	F
0	0	0
0	1	1
1	0	1
1	1	0

Logic symbol

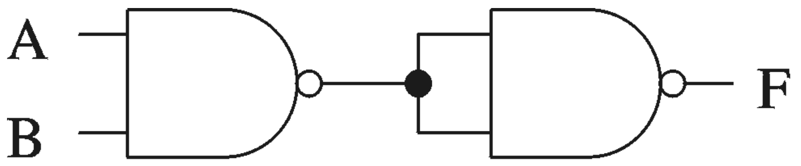
Truth table

Basic Concepts (cont.)

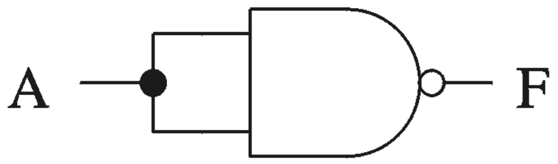
- Complete sets
  - A set of gates is complete
    - If we can implement any logical function using only the type of gates in the set
      - You can uses as many gates as you want
  - Some example complete sets
    - {AND, OR, NOT} Not a minimal complete set
    - {AND, NOT}
    - {OR, NOT} $\longleftarrow$
    - {NAND}
    - {NOR}
  - Minimal complete set
      - A complete set with no redundant elements.

Basic Concepts (cont.)

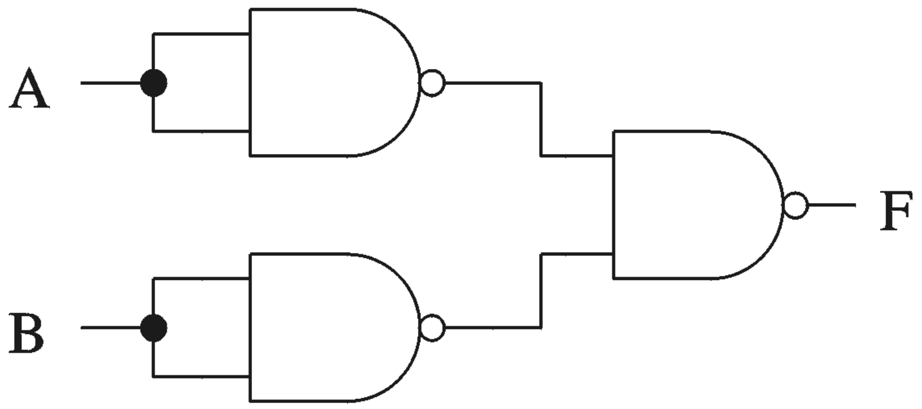
- Proving NAND gate is universal



AND gate



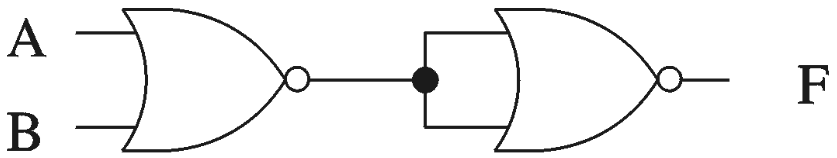
NOT gate



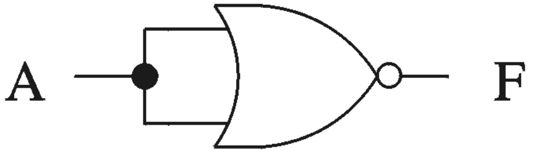
OR gate

Basic Concepts (cont.)

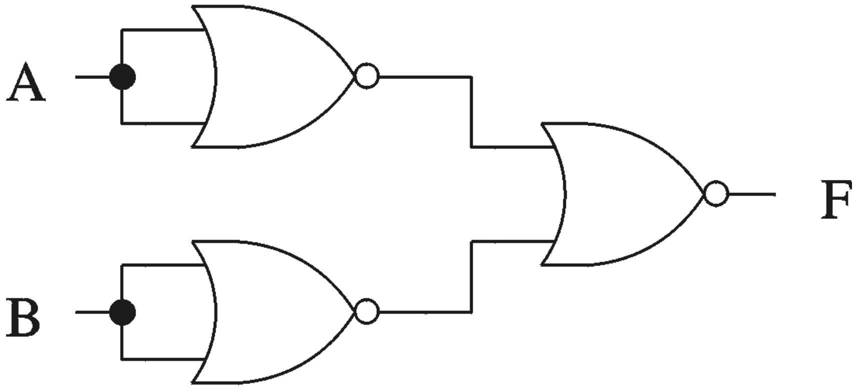
- Proving NOR gate is universal



OR gate

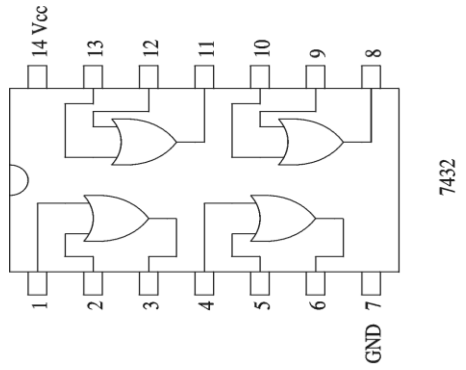


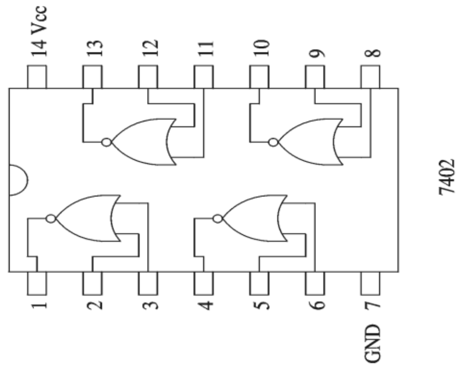
NOT gate

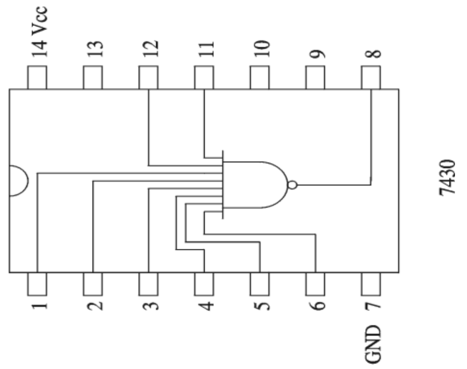


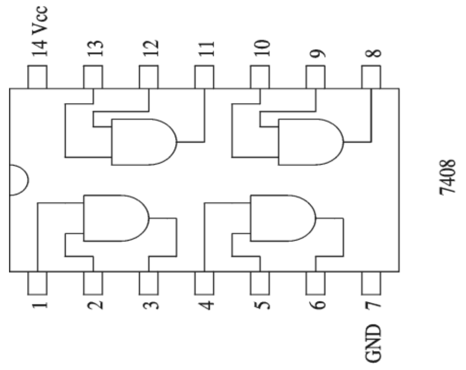
AND gate

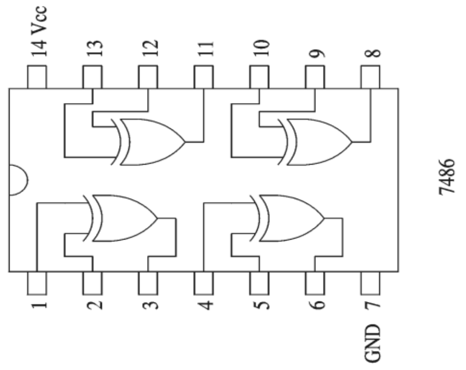
Logic Chips (cont.)

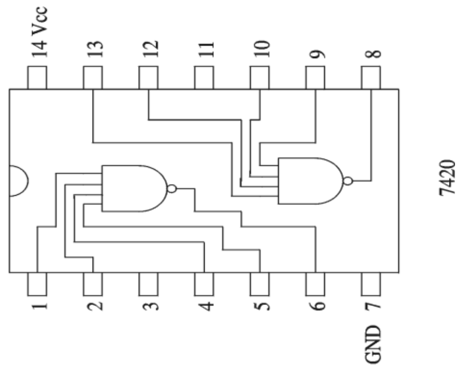


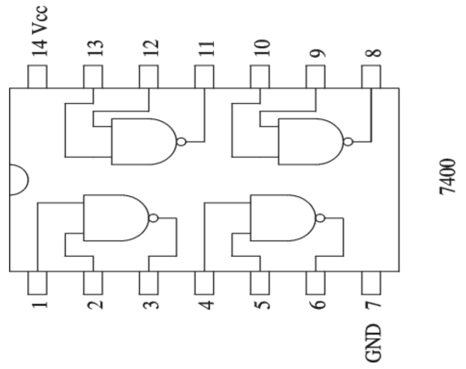


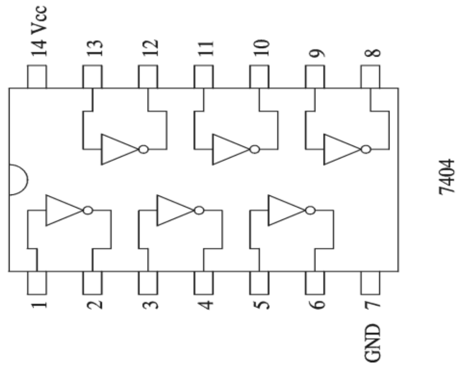










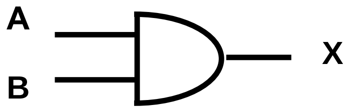


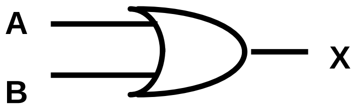
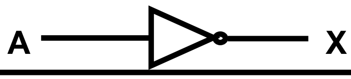
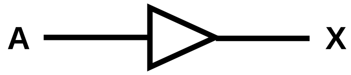
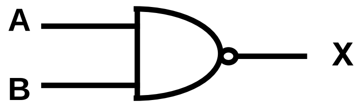
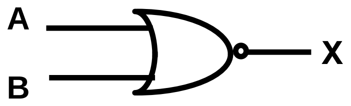
Logic Chips (cont.)

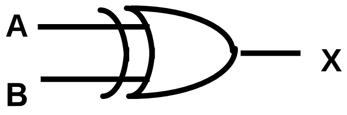
- Integration levels
  - SSI (small scale integration)
    - Introduced in late 1960s
    - 1-10 gates (previous examples)
  - MSI (medium scale integration)
    - Introduced in late 1960s
    - 10-100 gates
  - LSI (large scale integration)
    - Introduced in early 1970s
    - 100-10,000 gates
  - VLSI (very large scale integration)
    - Introduced in late 1970s
    - More than 10,000 gates

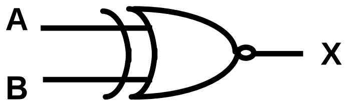
COMBINATIONAL GATES

Name	Symbol	Function	Truth Table

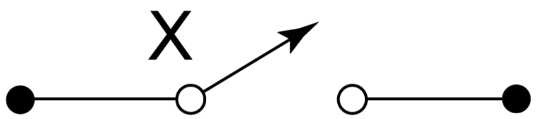
AND		X = A • B
		or
		X = AB
			A B X
			0 0 0
			0 1 0
			1 0 0
			1 1 1

OR		X = A + B	A B X 0 0 0 0 1 1 1 0 1 1 1 1
I		X = A’	A X 0 1 1 0
Buffer		X = A	A X 0 0 1 1
NAND		X = (AB)’	A B X 0 0 1 0 1 1 1 0 1 1 1 0
NOR		X = (A + B)’	A B X 0 0 1 0 1 0 1 0 0 1 1 0

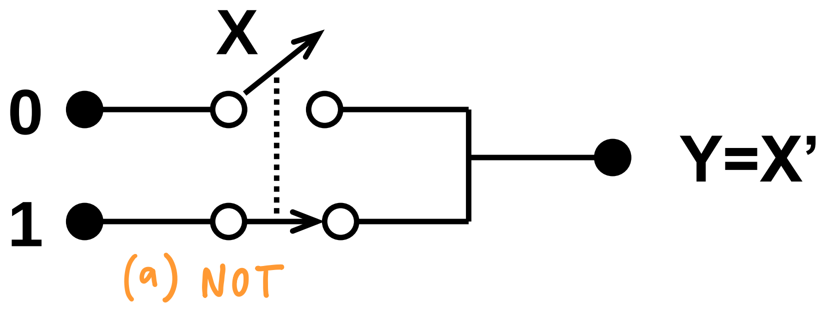
XOR
Exclusive OR
		X = A ⊕ B
		or
		X = A’B + AB’
			A B X
			0 0 0
			0 1 1
			1 0 1
			1 1 0

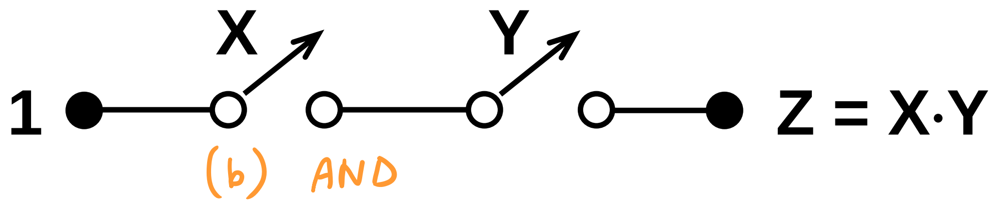
XNOR
Exclusive NOR
or Equivalence
		X = (A ⊕ B)’
		or
		X = A’B’+ AB
			A B X
			0 0 1
			0 1 0
			1 0 0
			1 1 1

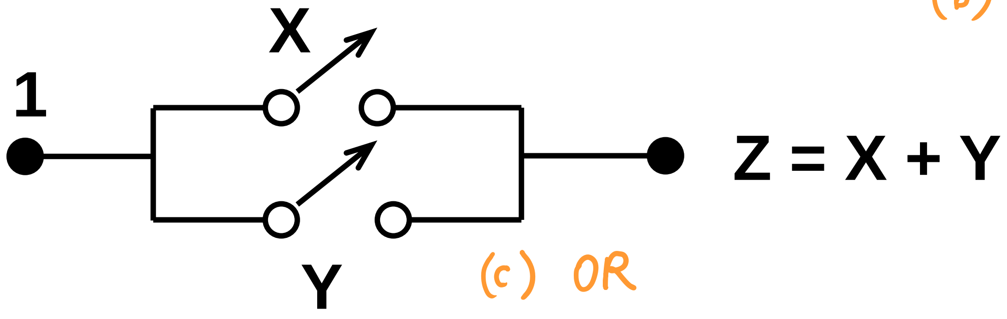
NOT, AND, OR as switches



X = 0 switch open
X = 1 switch closed

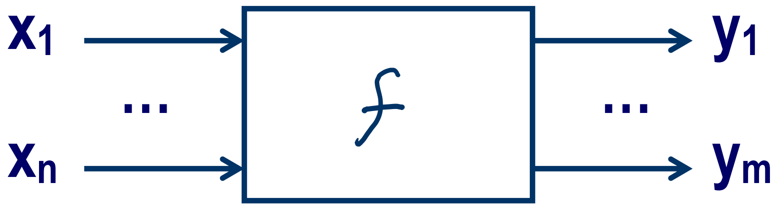






Combinational Logic

- Logic systems are classified into two types:
  - Combinational
  - Sequential

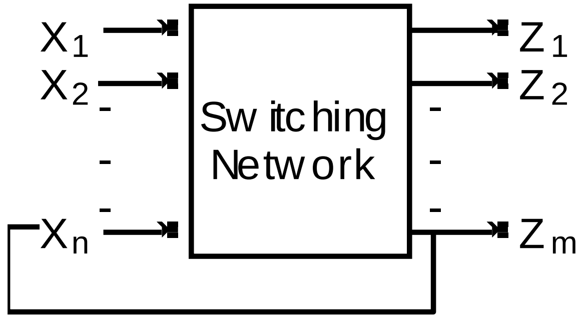


- A combinational logic system is one whose current outputs depend only on its current inputs

- Combinational systems are memory-less. They do contain feedback loops.
  - A feedback loop is a signal path that allows the output signal of a system to propagate back to the input of the system.

Digital Hardware Systems

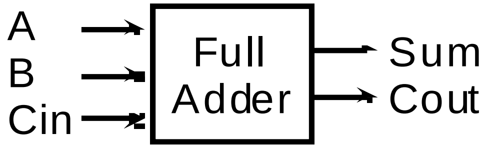
Combinational vs. Sequential Logic



Network implemented from switching elements or logic gates. The presence of feedback distinguishes between *sequential* and *combinational* networks.

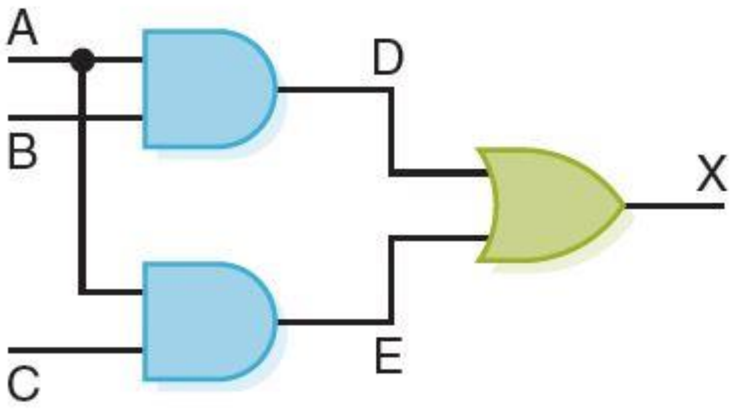
Combinational logic

no feedback among inputs and outputs
outputs are a pure function of the inputs
e.g., full adder circuit:
(A, B, Carry In) mapped into (Sum, Carry Out)



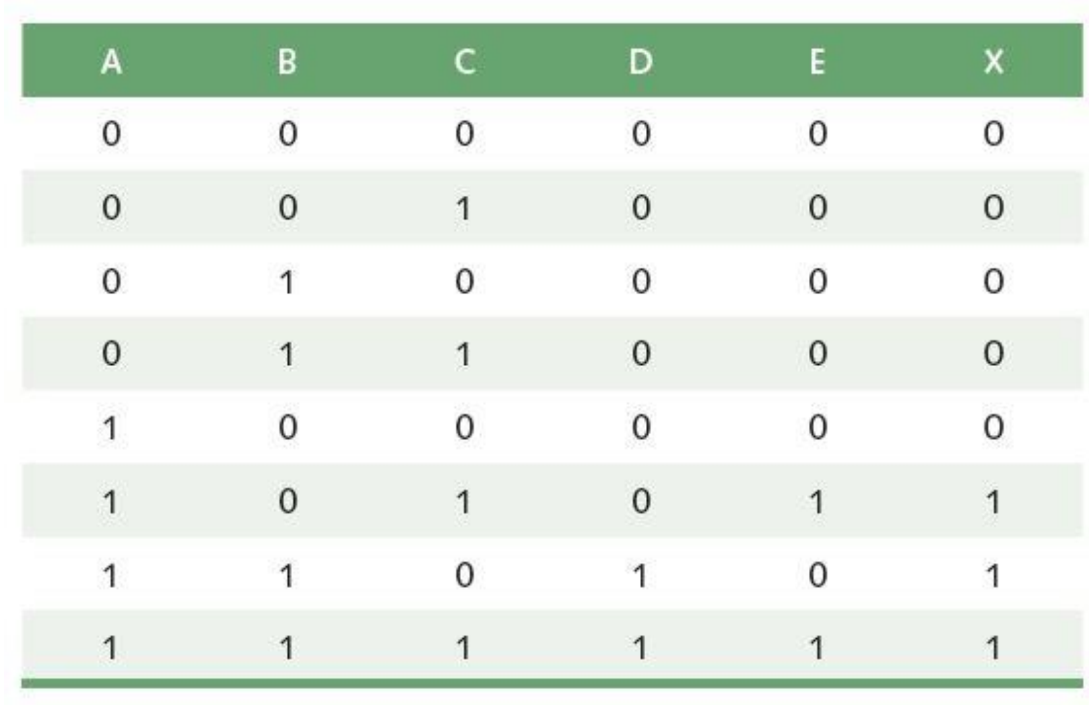
Combinational Circuits

Gates are combined into circuits by using the output of one gate as the input for another



This same circuit using a Boolean expression is AB + AC

Combinational Circuits

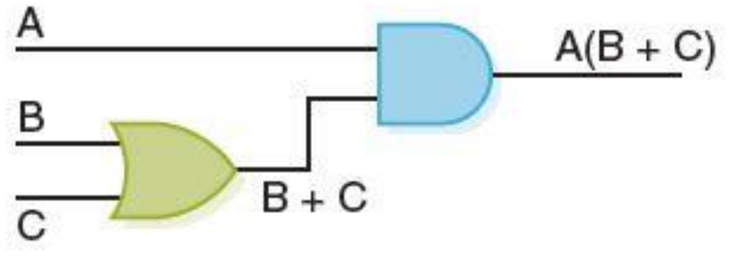


A	B	C	D	E	X
0	0	0	0	0	0
0	0	1	0	0	0
0	1	0	0	0	0
0	1	1	0	0	0
1	0	0	0	0	0
1	0	1	0	1	1
1	1	0	1	0	1
1	1	1	1	1	1

Three inputs require eight rows to describe all possible input combinations

Combinational Circuits

Consider the following Boolean expression A(B + C)



A	B	C	B + C	A(B + C)
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	1	0
1	0	0	0	0
1	0	1	1	1
1	1	0	1	1
1	1	1	1	1

Does this truth table look familiar?

Compare it with previous table